Claims:

1. A method for storing data in a memory in a computer system, the method comprising:

receiving uncompressed data;

determining a compression mode for the data, wherein the compression mode comprises one of lossless compression, lossy compression, or no compression;

selectively compressing the uncompressed data, wherein said compressing is selectively performed in response to the compression mode for the data; and

storing the data in the memory.

2. The method of claim 1,

wherein, in said selectively compressing:

the data is compressed using a lossless compression format if said compression mode indicates lossless compression for the data;

the data is compressed using a lossy compression format if said compression mode indicates lossy compression for the data; and

the data is not compressed if said compression mode indicates no compression for the data.

3. The method of claim 1,

wherein the data is stored in the memory in a lossless compression format if said compression mode indicates lossless compression for the data;

wherein the data is stored in the memory in a lossy compression format if said compression mode indicates lossy compression for the data;

wherein the data is stored in the memory in an uncompressed format if said compression mode indicates no compression for the data.

4. The method of claim 1,

wherein the compression mode is determined in response to one or more of: a requesting agent which provides the data; an address range where the data is stored; and/or a data type of the data.

5. The method of claim 1, further comprising:

receiving one or more destination addresses indicating a storage destination for the data in the memory;

wherein said determining the compression mode comprises analyzing the one or more destination addresses to determine the compression mode.

6. The method of claim 5, wherein the memory includes a first address range designated with a first compression format, and a second address range designated with a second compression format;

wherein said determining the compression mode comprises analyzing the one or more destination addresses to determine if the one or more destination addresses reside in the first address range or the second address range.

7. The method of claim 5, wherein the memory includes a first address range designated with a lossless compression format, and a second address range designated with a lossy compression format;

wherein said determining the compression mode comprises analyzing the one or more destination addresses to determine if the one or more destination addresses reside in the first address range or the second address range.

8. The method of claim 5, wherein the memory includes a first address range designated with a lossless compression format, a second address range designated with a lossy compression format, and a third address range designated with a no compression format;

wherein said determining the compression mode comprises analyzing the one or more destination addresses to determine if the one or more destination addresses reside in the first address range, the second address range, or the third address range.

9. The method of claim 1, further comprising:

wherein the uncompressed data is received from a requesting agent; and

wherein said determining the compression mode for the data comprises determining the compression mode based on the requesting agent.

10. The method of claim 1, wherein the requesting agent is one of a CPU application or a video/graphics driver.

11. The method of claim 1, wherein the data has a data type

wherein said determining the compression mode for the data comprises determining the compression mode based on the data type of the data.

12. The method of claim 11, wherein the data comprises one of application data or video/graphics data;

wherein the compression mode is determined to be lossless compression if the data comprises application data; and

wherein the compression mode is determined to be lossy compression if the data comprises video/graphics data.

13. The method of claim 1, wherein the computer system includes a CPU, wherein the memory comprises system memory which stores application code and data executed by the CPU.

14. The method of claim 1, wherein the computer system further includes a memory controller which controls operation of the system memory, wherein the memory controller includes a compression/decompression engine;

wherein the memory controller implements the method.

15. The method of claim 1, further comprising:

receiving a request for the data;

accessing the data from the memory in response to the request;

determining a compression mode for the data in response to receiving the request;

selectively decompressing the data, wherein said decompressing is selectively performed in response to the compression mode for the data; and

providing the data in response to the request.

16. The method of claim 15,

wherein, in said selectively decompressing:

the data is decompressed if said compression mode indicates compression for the data; and

the data is not decompressed if said compression mode indicates no compression for the data.

17. The method of claim 15,

wherein, in said selectively decompressing:

the data is decompressed if said compression mode indicates the data is stored in the memory in a compressed format; and

the data is not decompressed if said compression mode indicates the data is stored in the memory in an uncompressed format.

18. The method of claim 15,

wherein, in said selectively decompressing:

the data is decompressed using lossless decompression if said compression mode indicates lossless compression for the data;

the data is decompressed using lossy decompression if said compression mode indicates lossy compression for the data; and

the data is not decompressed if said compression mode indicates no compression for the data.

19. The method of claim 1, wherein said storing the data in the memory includes storing compression mode information in the memory with the data;

wherein the compression mode information indicates a decompression procedure for decompression of the compressed first data.

20. The method of claim 19, wherein the compression mode information is embedded in the data.

21. The method of claim 19, wherein the compression mode information is stored in a non-compressed format regardless of the compression mode of the data.

22. The method of claim 19, further comprising:

receiving a request for the data;

analyzing the compression mode information to determine a compression mode for the data in response to receiving the request;

accessing the data from the memory in response to the request;

selectively decompressing the data, wherein said decompressing is selectively performed in response to the compression mode for the data; and

providing the data in response to the request.

23. The method of claim 22,

wherein, in said selectively decompressing:

the data is decompressed using lossless decompression if said compression mode information indicates lossless compression for the data;

the data is decompressed using lossy decompression if said compression mode information indicates lossy compression for the data; and

the data is not decompressed if said compression mode information indicates no compression for the data.

24. The method of claim 1, further comprising:

creating a header after said determining the compression mode for the data, wherein the header includes compression mode information indicating the compression mode of the first data, wherein the compression mode information indicates a decompression procedure for decompression of the compressed first data; and

wherein said storing the data in the memory includes storing the header in the memory with the data.

25. The method of claim 24, further comprising:

receiving a request for the data;

accessing the data from the memory in response to the request;

analyzing the header to determine a compression mode for the data in response to receiving the request;

selectively decompressing the data, wherein said decompressing is selectively performed in response to the compression mode for the data; and

providing the data in response to the request.

26. A computer system utilizing compressed storage of data, the computer system comprising:

a CPU;

system memory which stores data used by said CPU for executing one or more applications, wherein the system memory also stores an operating system;

a memory controller coupled to said system memory and said CPU, wherein said memory controller performs memory control functions for said system memory, wherein said memory controller includes a compression/decompression engine comprised in said memory controller for compressing and decompressing data transferred to or from said system memory;

wherein the memory controller is operable to:

receive uncompressed data;

determine a compression mode for the data, wherein the compression mode comprises one of lossless compression, lossy compression, or no compression;

selectively compress the uncompressed data, wherein said compressing is selectively performed in response to the compression mode for the data; and

store the data in the memory.

27. The computer system of claim 26,

wherein the compression mode is determined in response to one or more of: a requesting agent which provides the data; an address range where the data is stored; and/or a data type of the data.

28. The computer system of claim 26,

wherein the memory controller is operable to receive one or more destination addresses indicating a storage destination for the data in the memory;

wherein the memory controller is operable to analyze the one or more destination addresses to determine the compression mode.

29. The computer system of claim 26,

wherein the uncompressed data is received from a requesting agent; and

wherein the memory controller determines the compression mode based on the requesting agent.

30. The computer system of claim 26, wherein the data has a data type;

wherein the memory controller determines the compression mode for the data based on the data type of the data.

31. A method for compressing data and storing the compressed data in a memory in a computer system, the method comprising:

receiving uncompressed data;

receiving one or more destination addresses indicating a storage destination for the data in the memory;

determining a compression mode for the data based on the one or more destination addresses;

selectively compressing the uncompressed data, wherein said compressing is selectively performed in response to the compression mode for the data; and

storing the data in the memory at the one or more destination addresses.

32. The method of claim 31, wherein the compression mode comprises one of lossless compression, lossy compression, or no compression;

wherein, in said selectively compressing:

the data is compressed using a lossless compression format if said compression mode indicates lossless compression for the data;

the data is compressed using a lossy compression format if said compression mode indicates lossy compression for the data; and

the data is not compressed if said compression mode indicates no compression for the data.

33. A method for compressing data and storing the compressed data in a memory in a computer system, the method comprising:

receiving uncompressed data from a requesting agent;

determining a compression mode for the data based on the requesting agent;

selectively compressing the uncompressed data, wherein said compressing is selectively performed in response to the compression mode for the data; and

storing the data in the memory at the one or more destination addresses.

34. The method of claim 33, wherein the compression mode comprises one of lossless compression, lossy compression, or no compression;

wherein, in said selectively compressing:

the data is compressed using a lossless compression format if said compression mode indicates lossless compression for the data;

the data is compressed using a lossy compression format if said compression mode indicates lossy compression for the data; and

the data is not compressed if said compression mode indicates no compression for the data.

35. A method for compressing data and storing the compressed data in a memory in a computer system, the method comprising:

receiving uncompressed data, wherein the data has a data type;

determining a compression mode for the data based on the data type of the data;

selectively compressing the uncompressed data, wherein said compressing is selectively performed in response to the compression mode for the data; and

storing the data in the memory at the one or more destination addresses.

36. The method of claim 35, wherein the compression mode comprises one of lossless compression, lossy compression, or no compression;

wherein, in said selectively compressing:

the data is compressed using a lossless compression format if said compression mode indicates lossless compression for the data;

the data is compressed using a lossy compression format if said compression mode indicates lossy compression for the data; and

the data is not compressed if said compression mode indicates no compression for the data.

Sub B1 > 37. A method for storing data in a memory in a computer system, the method comprising:

receiving uncompressed data;

determining a compression mode for the data;

selectively compressing the uncompressed data, wherein said compressing is selectively performed in response to the compression mode for the data; and

storing the data in the memory.

38. The method of claim 37,

wherein, in said selectively compressing:

the data is compressed if said compression mode indicates compression for the data; and

the data is not compressed if said compression mode indicates no compression for the data;

wherein the data is stored in the memory in a compressed format if said compression mode indicates compression for the data;

wherein the data is stored in the memory in an uncompressed format if said compression mode indicates no compression for the data.

39. The method of claim 37,

wherein the compression mode is determined in response to one or more of a requesting agent which provides the data; an address range where the data is stored; and/or a data type of the data.

40. The method of claim 37, further comprising:

receiving one or more destination addresses indicating a storage destination for the data in the memory;

wherein said determining the compression mode comprises analyzing the one or more destination addresses to determine the compression mode.

41. The method of claim 37, further comprising:

wherein the uncompressed data is received from a requesting agent; and

wherein said determining the compression mode for the data comprises determining the compression mode based on the requesting agent.

42. The method of claim 37, wherein the data has a data type;

wherein said determining the compression mode for the data comprises determining the compression mode based on the data type of the data.

43. The method of claim 37, wherein the computer system includes a CPU, wherein the memory comprises system memory which stores application code and data executed by the CPU.

44. The method of claim 43, wherein the computer system further includes a memory controller which controls operation of the system memory, wherein the memory controller includes a compression/decompression engine;

wherein the memory controller implements the method.

45. The method of claim 37, further comprising:

receiving a request for the data;

accessing the data from the memory in response to the request;

determining a compression mode for the data in response to receiving the request;

selectively decompressing the data, wherein said decompressing is selectively performed in response to the compression mode for the data; and

providing the data in response to the request.

46. The method of claim 45,

wherein, in said selectively decompressing:

the data is decompressed if said compression mode indicates compression for the data; and

the data is not decompressed if said compression mode indicates no compression for the data.

47. A method for compressing data and storing the compressed data in a memory in a computer system, the method comprising:

receiving uncompressed first data;

selectively compressing the uncompressed first data to produce compressed first data according to a compression mode;

creating a header, wherein the header includes compression mode information indicating the compression mode of the first data, wherein the compression mode information indicates a decompression procedure for decompression of the compressed first data; and

storing the compressed first data and the header in the memory.

48. The method of claim 47, wherein the compression mode information is used to select a decompression process which is applied to the compressed first data.

49. The method of claim 47, wherein the compression mode information indicates one of: lossless compression or lossy compression.

50. The method of claim 47, wherein the compression mode information indicates one of: lossless compression, lossy compression, or no compression.

51. The method of claim 47, wherein the header is embedded in the compressed first data.

52. The method of claim 47, further comprising:

decompressing the compressed first data to produce uncompressed first data, wherein said decompressing comprises:

analyzing the compression mode information comprised in the header;

determining a decompression procedure based on the compression mode information comprised in the header; and

decompressing the compressed first data using the determined decompression procedure.

53. The method of claim 47, further comprising:

determining the compression mode prior to compressing the uncompressed first data.

54. A method for compressing data and storing the compressed data in a memory in a computer system, the method comprising:

receiving uncompressed first data;

selectively compressing the uncompressed first data to produce compressed first data according to a compression mode;

storing the compressed first data and the header in the memory, wherein said storing the data in the memory includes storing compression mode information in the memory with the data, wherein the compression mode information indicates a decompression procedure for decompression of the compressed first data.

55. The method of claim 54, wherein the compression mode information indicates one of: lossless compression or lossy compression.

56. The method of claim 54, wherein the compression mode information indicates one of: lossless compression, lossy compression, or no compression.

57. The method of claim 54, further comprising:

decompressing the compressed first data to produce uncompressed first data, wherein said decompressing comprises:

analyzing the compression mode information stored with the data;

determining a decompression procedure based on the compression mode information stored with the data; and

decompressing the compressed first data using the determined decompression procedure.

58. A method for compressing data and storing the compressed data in a memory in a computer system, the method comprising:

receiving uncompressed first data;

compressing the uncompressed first data to produce compressed first data, wherein said compressed first data has a first size;

determining if the first size of the compressed first data is greater than an allocated memory block size of a first allocated memory block;

creating a header, wherein the header includes an overflow indicator indicating whether the first size of the compressed first data is greater than the allocated memory block size; and

storing the compressed first data and the header in the memory.

59. The method of claim 58, wherein said determining determines that the first size of the compressed first data is less than or equal to the allocated memory block size;

wherein the overflow indicator indicates that the first allocated memory block stores all of the compressed first data.

60. The method of claim 59, wherein said overflow indicator indicates that the last symbol of the compressed first data is stored in the first allocated memory block.

61. The method of claim 58, wherein said determining determines that the first size of the compressed first data is greater than the allocated memory block size;

wherein the overflow indicator indicates that the first allocated memory block does not store all of the compressed first data;

the method further comprising:

allocating a first overflow memory block;

storing overflow information in the header, wherein the overflow information includes an overflow address pointer which points to the first overflow memory block;

wherein said storing comprises:

storing a first portion of the compressed first data and the header in the first allocated memory block; and

storing an overflow portion of the compressed first data in the first overflow memory block.

62. The method of claim 61, wherein the first overflow memory block has a fixed size.

63. The method of claim 61, further comprising:

determining whether the overflow portion has a size greater than the first overflow memory block;

creating an overflow header, wherein the overflow header includes an overflow indicator indicating whether the overflow portion has a size greater than the first overflow memory block;

wherein said storing the overflow portion includes storing the overflow portion and the overflow header in the first overflow memory block.

64. The method of claim 63, further comprising:

wherein said determining determines that the overflow portion of the compressed first data has a size greater than the first overflow memory block;

wherein the overflow indicator in the overflow header indicates that the first overflow memory block does not store all of the overflow portion;

the method further comprising:

allocating a second overflow memory block in response to determining that the overflow portion of the compressed first data is greater than the first overflow memory block;

storing overflow information in the first overflow header, wherein the overflow information includes an overflow address pointer which points to the second overflow memory block;

wherein said storing comprises:

storing a first portion of the compressed first data and the header in the first allocated memory block;

storing a first overflow portion of the compressed first data in the first overflow memory block; and

storing a second overflow portion of the compressed first data in the second overflow memory block.

65. The method of claim 58, wherein said determining determines that the first size of the compressed first data is greater than the allocated memory block size;

wherein the overflow indicator indicates that the first allocated memory block does not store all of the compressed first data;

the method further comprising:

allocating a plurality of overflow memory blocks, including a first overflow memory block and a last overflow memory block;

storing overflow information in the header, wherein the overflow information includes an overflow address pointer which points to a first overflow memory block;

wherein said storing comprises:

storing a first portion of the compressed first data and the header in the first allocated memory block; and

for each of the overflow memory blocks except the last overflow memory block, storing, in the respective overflow memory block, an overflow portion of the compressed first data and a header pointing to a subsequent overflow memory block.

66. The method of claim 58, wherein said determining determines that the first size of the compressed first data is greater than the allocated memory block size;

wherein the overflow indicator indicates that the first allocated memory block does not store all of the compressed first data;

the method further comprising:

allocating one or more overflow memory blocks, wherein the first allocated memory block and the one or more overflow memory blocks are insufficient to store the compressed first data;

generating an interrupt to a driver in response to the first allocated memory block and the one or more overflow memory blocks being insufficient to store the compressed first data;

the driver allocating additional overflow memory blocks in response to the interrupt.

67. The method of claim 58, wherein said determining determines if the first size of the compressed first data and a maximum header size are greater than the allocated memory block size.

68. The method of claim 58, further comprising:

allocating the first allocated memory block in response to receiving the uncompressed first data, wherein the first allocated memory block is allocated according to a pre-determined compression ratio.

69. The method of claim 58, wherein the computer system includes an operating system, the method further comprising:

the operating system allocating the first allocated memory block in response to receiving the uncompressed first data.

70. A computer system including a memory controller having an embedded compression/decompression engine, the computer system comprising:

a CPU;

system memory which stores data used by said CPU for executing one or more applications;

a memory controller coupled to said system memory and said CPU, wherein said memory controller performs memory control functions for said system memory, wherein said memory controller includes said compression/decompression engine comprised in said memory controller for compressing and decompressing data transferred to or from said system memory;

wherein said memory controller is operable to:

receive uncompressed first data;

selectively compress the uncompressed first data to produce compressed first data according to a compression mode;

create a header, wherein the header includes compression mode information indicating the compression mode of the first data, wherein the compression mode information indicates a decompression procedure for decompression of the compressed first data; and

store the compressed first data and the header in the memory.

71. A method for performing parallel compression of data, wherein the method maintains a history table comprising entries, wherein each entry comprises one symbol, the method comprising:

a) receiving uncompressed data, wherein the uncompressed data comprises a plurality of symbols;

b) comparing a plurality of symbols with each entry in a history table in a parallel fashion, wherein said comparing produces compare results;

wherein the method maintains a current count of prior matches which occurred when previous symbols were compared with entries in the history table, wherein a count is maintained for each entry in the history table;

c) determining match information for each of said plurality of symbols based on the current count and the compare results; and

d) outputting compressed data in response to the match information.

72. The method of claim 71, wherein said outputting compressed data includes:

outputting a count value and an entry pointer for a contiguous match, wherein the entry pointer points to the entry in the history table which produced the contiguous match, wherein the count value indicates a number of matching symbols in the contiguous match.

73. The method of claim 72, wherein said outputting the count value includes encoding a value representing the count value; wherein more often occurring counts are encoded with fewer bits than less often occurring counts.

74. The method of claim 72, wherein said outputting compressed data further includes:

for non-matching symbols which do not match any entry in the history table, outputting the non-matching symbols.

75. The method of claim 71, further comprising:

e) repeating steps a) – d) one or more times until no more data is available; and

f) when no more data is available, if any current counts are non-zero, outputting compressed data for the longest remaining match in the history table.

76. The method of claim 71, wherein said determining match information includes determining zero or more matches of said plurality of symbols with each entry in the history table.

77. The method of claim 71, wherein said determining match information includes:

resetting the counts for all entries if the compare results indicate a contiguous match did not match one of the plurality of symbols.

78. The method of claim 71, wherein the counts for all entries are reset based on the number of the plurality of symbols that did not match in the contiguous match.

79. The method of claim 71, wherein said determining match information includes:

generating a reset value for all entries based on the compare results for a contiguous match, wherein the reset value indicates a number of the plurality of symbols that did not match in the contiguous match as indicated in the compare results; and

updating the current count according to the compare results and the reset value.

80. The method of claim 71, wherein said determining match information includes:

determining a longest contiguous match based on the current count and the compare results;

determining if the longest contiguous match has stopped matching;

if the longest contiguous match has stopped matching, then:

generating a reset value for all entries based on the compare results for the longest contiguous match, wherein the reset value indicates a number of the plurality of symbols that did not match in the longest contiguous match as indicated in the compare results;

updating the current count according to the compare results and the reset value; and

wherein said outputting compressed data includes outputting compressed data corresponding to the longest contiguous match;

81. The method of claim 80, wherein said outputting compressed data corresponding to the longest contiguous match comprises outputting a count value and an entry pointer, wherein the entry pointer points to the entry in the history table which produced the longest contiguous match, wherein the count value indicates a number of matching symbols in the longest contiguous match.

82. The method of claim 71, wherein the plurality of symbols includes a first symbol, a last symbol, and one or more middle symbols;

wherein said determining match information includes:

if at least one contiguous match occurs with one or more respective contiguous middle symbols, and the one or more respective contiguous middle symbols are not involved in a match with either the symbol before or after the respective contiguous middle symbols, then:

selecting the one or more largest non-overlapping contiguous matches involving the middle symbols;

wherein said outputting compressed data includes outputting compressed data for each of the selected matches involving the middle symbols.

83. The method of claim 71, wherein said determining match information and said outputting compressed data in response to the match information comprises:

determining zero or more matches of said plurality of symbols with each entry in the history table;

examining the compare results for each entry;

for non-matching symbols which do not match any entry in the history table, outputting the non-matching symbols;

if any entry stopped matching, examining current counts and the compare results for every entry;

determining the longest contiguous match based on the current count and the compare results;

determining if the longest contiguous match has stopped matching;

if the longest contiguous match has stopped matching, then:

outputting a count value and an entry pointer, wherein the entry pointer points to the entry in the history table which produced the longest contiguous match, wherein the count value indicates a number of matching symbols in the longest contiguous match; and

generating a reset value for all entries based on the compare results for the longest match, wherein the reset value indicates a number of the plurality of symbols that did not match in the longest contiguous match as indicated in the compare results;

updating the current count according to the compare results and the reset value;

the method further comprising:

e) repeating steps a) – d) one or more times until no more data is available; and

f) when no more data is available, if any current counts are non-zero, outputting a count value and an entry pointer for the longest remaining match in the history table.

84. The method of claim 83, wherein the plurality of symbols includes a first symbol, a last symbol, and one or more middle symbols;

wherein, if the longest contiguous match has stopped matching, then the method further comprises:

if at least one contiguous match occurs with one or more respective contiguous middle symbols, and the one or more respective contiguous middle symbols are not involved in a match with either the symbol before or after the respective contiguous middle symbols, then:

selecting the largest non-overlapping contiguous matches involving the middle symbols;

outputting a count value and an entry pointer for each of the selected matches involving the middle symbols.

85. The method of claim 71, wherein the plurality of symbols comprise a power of 2 number of symbols.

86. The method of claim 71, wherein the plurality of symbols comprise 4 symbols.

87. A method for performing parallel compression of data, wherein the method maintains a history table comprising entries, wherein each entry comprises one symbol, the method comprising:

a) receiving uncompressed data, wherein the uncompressed data comprises a plurality of symbols;

b) comparing a plurality of symbols with each entry in a history table in a parallel fashion, wherein said comparing produces compare results;

wherein the method maintains a current count of prior matches which occurred when previous symbols were compared with entries in the history table, wherein a count is maintained for each entry in the history table;

c) determining zero or more matches of said plurality of symbols with each entry in the history table;

d) for non-matching symbols which do not match any entry in the history table, then outputting the non-matching symbols;

e) if any entry stopped matching, then examining current counts and the compare results for every entry;

f) determining the longest contiguous match based on the current count and the compare results;

g) determining if the longest contiguous match has stopped matching;

h) if the longest contiguous match has stopped matching, then:

outputting a count value and an entry pointer, wherein the entry pointer points to the entry in the history table which produced the longest contiguous match, wherein the count value indicates a number of matching symbols in the longest contiguous match;

generating a reset value for all entries based on the compare results for the longest contiguous match, wherein the reset value indicates a number of the plurality of symbols that did not match in the longest contiguous match as indicated in the compare results;

i) updating the current count according to the compare results and the reset value;

j) repeating steps a) – i) one or more times until no more data is available;

k) when no more data is available, if any current counts are non-zero, outputting a count value and an entry pointer for the longest remaining match in the history table.

88. The method of claim 87, wherein the plurality of symbols includes a first symbol, a last symbol, and one or more middle symbols;

wherein, if the longest contiguous match has stopped matching in step h), then in step h) the method further comprises:

if at least one contiguous match occurs with one or more respective contiguous middle symbols, but the one or more respective contiguous middle symbols are not involved in a match with either the symbol before or after the respective contiguous middle symbols, then:

selecting the largest non-overlapping contiguous matches involving the middle symbols;

outputting a count value and an entry pointer for each of the selected matches involving the middle symbols.

89. The method of claim 87, wherein in step d) the outputting the non-matching symbols includes outputting a flag indicating the output non-matching symbols are symbols.

90. The method of claim 87, wherein said outputting the count value and the entry pointer for a match includes outputting a flag indicating the subsequent data comprises the count value and the entry pointer.

91. The method of claim 87, wherein said outputting the count value includes encoding a value representing the count value; wherein more often occurring counts are encoded with fewer bits than less often occurring counts.

92. The method of claim 87, wherein the plurality of symbols comprise a power of 2 number of symbols.

93. The method of claim 87, wherein the plurality of symbols comprise 4 symbols.

94. A method for performing parallel compression of data, wherein the method maintains a history window comprising entries, wherein each entry comprises one symbol, the method comprising:

a) receiving uncompressed data, wherein the uncompressed data comprises a plurality of symbols;

b) comparing a plurality of symbols with each entry in the history window in a parallel fashion, wherein said comparing produces compare results;

wherein the method maintains a current count of prior matches which occurred when previous symbols were compared with entries in the history window, wherein a count is maintained for each entry in the history window;

c) determining zero or more matches of said plurality of symbols with each entry in the history window;

d) if no matches exist for any of said plurality of symbols, then:

outputting compressed data information for any previous matches;

outputting the non-matching symbols;

resetting all counters to zero;

e) if all of the plurality of symbols match for a respective entry, then:

increasing the counter by the plurality of symbols;

f) if one or more, but not all, of the plurality of symbols match for a respective entry, then

f1) if one or more previous matches exist, then:

determining the longest contiguous match, including the one or more previous matches, based on the current count and the compare results; and

outputting previous compressed data information for the longest contiguous match;

f2) for each symbol, in parallel,

determining if the respective symbol is included in any match;

if the respective symbol is not included in any match, then outputting the symbol uncompressed;

if the respective symbol is included in any match, then:

determining if the match includes the last symbol;

outputting compressed data information for the match if the match does not include the last symbol; and

resetting counters to the maximum of the symbol count in the match if the match includes the last symbol;

g) after either of steps d), e) or f), adding the uncompressed plurality of symbols to the history window;

h) repeating steps a) – g) one or more times until no more data is available; and

i) when no more data is available, if any current counts are non-zero, outputting compressed data information for the longest remaining match in the history window.

95. A method for compressing data and storing the compressed data in a memory in a computer system, the method comprising:

allocating a memory block, wherein the memory block is allocated for uncompressed data;

receiving uncompressed first data;

receiving one or more destination addresses indicating a storage destination of the first data in the allocated memory block;

compressing the uncompressed first data to produce compressed first data;

storing the compressed first data in the allocated memory block at the one or more destination addresses.

96. The method of claim 95, wherein said storing does not perform address translation of the one or more destination addresses, wherein said storing provides reduced latency.

97. The method of claim 95, wherein the uncompressed first data has a first size, wherein the compressed first data has a second smaller size;

wherein said storing does not perform address translation of the one or more destination addresses, wherein said storing does not perform memory minimization.

98. The method of claim 95, wherein the computer system includes an operating system, wherein the operating system allocates the memory block for uncompressed data;

wherein the operating system does not account for the compression operation.

99. The method of claim 95, wherein the computer system includes an operating system, wherein the operating system allocates the memory block for uncompressed data;

wherein the operating system is unaware of the compression operation.

100. The method of claim 95, wherein the compressed first data occupies a first portion of the allocated memory block, the method further comprising:

allocating a portion of the allocated memory block as overflow storage.

101. The method of claim 100,

wherein the uncompressed first data comprises a plurality of blocks each having an original size, wherein one or more of the blocks compress to a larger size than the original size;

wherein said storing the compressed first data includes storing overflow data in the portion of the allocated memory block allocated as overflow storage.

102. The method of claim 95, wherein the uncompressed first data comprises application data generated by a CPU in the computer system.

103. The method of claim 95, wherein the memory comprises a system memory which stores application data generated by a CPU in the computer system.

104. The method of claim 95, further comprising

receiving a request for the first data;

decompressing the compressed first data to produce uncompressed first data;

providing the uncompressed first data in response to the request.

105. The method of claim 95, further comprising

receiving a request for the first data, wherein the request includes the one or more destination addresses in the allocated memory block where the compressed first data is stored;

accessing the compressed first data from the memory using the one or more destination addresses;

decompressing the compressed first data to produce uncompressed first data; and

providing the uncompressed first data in response to the request.

106. The method of claim 95, wherein the computer system includes a memory controller, wherein the memory controller performs said receiving uncompressed first data, said receiving one or more destination addresses, said compressing the uncompressed first data to produce compressed first data, and said storing the compressed first data.

107. A method for compressing data and storing the compressed data in a memory in a computer system, wherein the computer system includes an operating system, the method comprising:

the operating system allocating a memory block, wherein the operating system allocates the memory block for uncompressed data;

receiving uncompressed first data;

receiving one or more destination addresses indicating a storage destination of the first data in the allocated memory block;

compressing the uncompressed first data to produce compressed first data;

storing the compressed first data in the allocated memory block at the one or more destination addresses, wherein said storing does not perform address translation of the one or more destination addresses for reduced latency;

wherein the operating system does not account for the compression operation.

108. The method of claim 107, wherein the uncompressed first data has a first size, wherein the compressed first data has a second smaller size;

wherein said storing does not perform address translation of the one or more destination addresses, wherein said storing does not perform memory minimization.

109. A computer system utilizing compressed storage of data, the computer system comprising:

a CPU;

system memory which stores data used by said CPU for executing one or more applications, wherein the system memory also stores an operating system;

a memory controller coupled to said system memory and said CPU, wherein said memory controller performs memory control functions for said system memory, wherein said memory controller includes a compression/decompression engine comprised in said memory controller for compressing and decompressing data transferred to or from said system memory;

wherein memory blocks are allocated in the system memory for uncompressed data;

wherein the memory controller is operable to:

receive uncompressed first data;

receive one or more destination addresses indicating a storage destination of the first data in an allocated memory block;

compress the uncompressed first data to produce compressed first data; and

store the compressed first data in the allocated memory block at the one or more destination addresses.

110. The computer system of claim 109, wherein, in storing the compressed first data, the memory controller does not perform address translation of the one or more destination addresses, wherein the memory controller provides reduced latency.

111. The computer system of claim 109, wherein the uncompressed first data has a first size, wherein the compressed first data has a second smaller size;

wherein the memory controller does not perform address translation of the one or more destination addresses, wherein the memory controller does not perform memory minimization.

112. The computer system of claim 109, wherein the computer system includes an operating system, wherein the operating system allocates the memory block for uncompressed data;

wherein the operating system does not account for the compression operation.

113. The computer system of claim 109, wherein the computer system includes an operating system, wherein the operating system allocates the memory block for uncompressed data;

wherein the operating system is unaware of the compression operation.

114. The computer system of claim 109, wherein the compressed first data occupies a first portion of the allocated memory block;

wherein the memory controller is operable to allocate a portion of the allocated memory block as overflow storage.

115. The computer system of claim 114,

wherein the uncompressed first data comprises a plurality of blocks each having an original size, wherein one or more of the blocks compress to a larger size than the original size;

wherein the memory controller is operable to store overflow data in the portion of the allocated memory block allocated as overflow storage.

116. The computer system of claim 109, wherein the uncompressed first data comprises application data generated by the CPU.

117. The computer system of claim 109, wherein the memory controller is further operable to:

receive a request for the first data;

decompress the compressed first data to produce uncompressed first data; and

provide the uncompressed first data in response to the request.

118. The computer system of claim 109, wherein the memory controller is further operable to:

receive a request for the first data, wherein the request includes the one or more destination addresses in the allocated memory block where the compressed first data is stored;

access the compressed first data from the system memory using the one or more destination addresses;

decompress the compressed first data to produce uncompressed first data; and

provide the uncompressed first data in response to the request.

119. A method for compressing data and storing the compressed data in a memory in a computer system, the method comprising:

allocating a memory block, wherein the memory block is allocated according to a pre-determined compression ratio;

receiving uncompressed first data;

receiving one or more destination addresses indicating a storage destination of the first data in the allocated memory block;

compressing the uncompressed first data to produce compressed first data; and

storing the compressed first data in the allocated memory block at the one or more destination addresses.

120. The method of claim 119, wherein said storing includes performing address translation of the one or more destination addresses, wherein said address translation minimizes memory usage.

121. The method of claim 119, wherein the computer system includes an operating system, wherein the operating system allocates the memory block for uncompressed data according to the pre-determined compression ratio.

122. The method of claim 119, wherein the uncompressed first data has a first size, wherein the compressed first data has a second smaller size;

the method further comprising:

determining if the compressed first data fits within the allocated memory block; and

allocating an overflow memory block if the compressed first data does not fit within the allocated memory block.